Data Management For Machine Learning

BITS Pilani

Data Management For Machine Learning
Lecture No.L4-Data Pipelines

Agenda

- Introduction
- Modes of Data Flow
- Modern Data Stack
- Data Pipelines
- Data Pipeline Architecture
- Data Pipeline Implementation
- Data Pipeline - Tools / Platforms

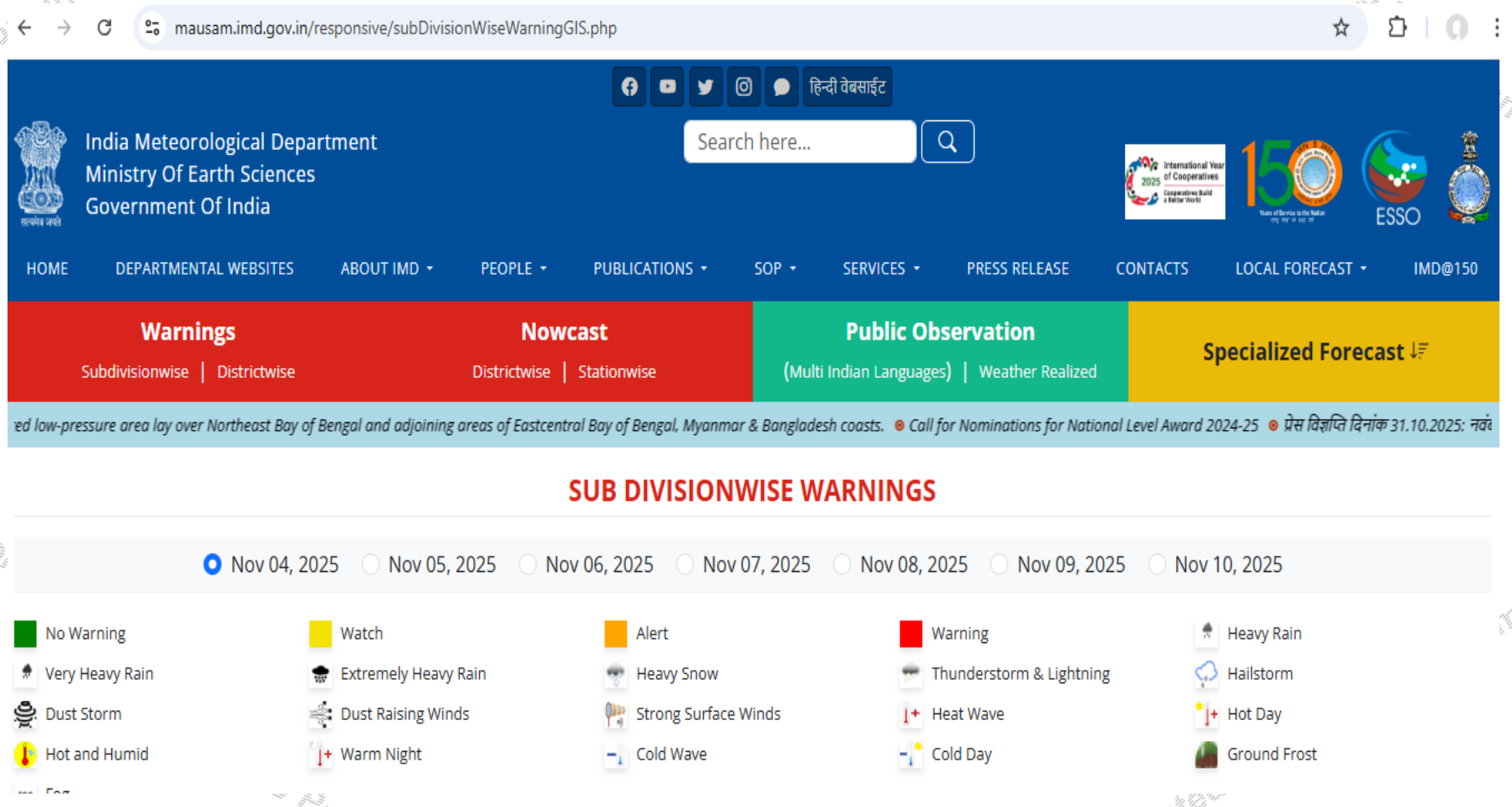
Work Integrated Learning Programmes

innovate achieve lead

BITS Pilani

Introduction

Have you seen this?



IMD source systems

Do you know from how many systems IMD collects data? Any guess?

Per https://mausam.imd.gov.in/event/evolution.pdf

- **806 Automatic Weather Stations (AWS)**
- **1,382 Automatic Rain Gauges (ARG)**
- **5,896 District-wise Rainfall Monitoring Scheme (DRMS) stations**
- **200 Agro-Automated Weather Stations (Agro-AWS)** and other Agromet observatories
- **56 Radiosonde/Radiowind (RS/RW) stations** for upper-air observations
- **39 Doppler Weather Radars (DWRs)**
- **Two geostationary satellites (INSAT-3D and INSAT-3DR)**, which a
  AWS/ARG network
- **Other specialized networks** for monitoring ozone (multiple statio
  chemistry (11 stations), aerosols (12 locations), black carbon (16 stati
  sensors).
- **Numerous manual surface observatories** (around 530, some of which e
  departmental).
- **Other systems** like High Wind Speed Recorders, Runway Visual Range (RVR) s stems at airports, and the System for Air Quality Forecasting And Research (SAFAR) stations in major cities

Why you need seamless flow of information? This massive system now can say

Can we do it manually multiple times everyday?

Why?

- Is it humanly possible to feed data?
- Why to do you need a data pipeline then?

well-designed pipeline is critical for handling the “5 V’s” of big data (volume, velocity, variety, veracity, value) while remaining efficient to maintain.

- Now is this reason justified for the previous example IMD?

- At the **heart of any data pipeline**, it’s fundamentally about **data flow** — how data moves **from one system to another**, often through a series of **processing, transformation, and storage stages**.

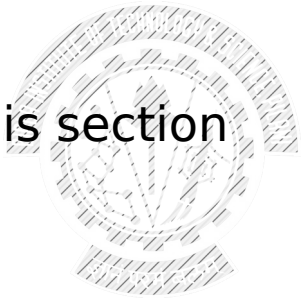
innovate achieve lead

BITS Pilani

Modes of Data Flow

Introduction

- In our previous sessions we had seen data flowing from one system to another, do you remember?
- Encoding, decoding, etc..
- Let us expand the same in this section



Work Integrated Learning Programmes

innovate achieve lead

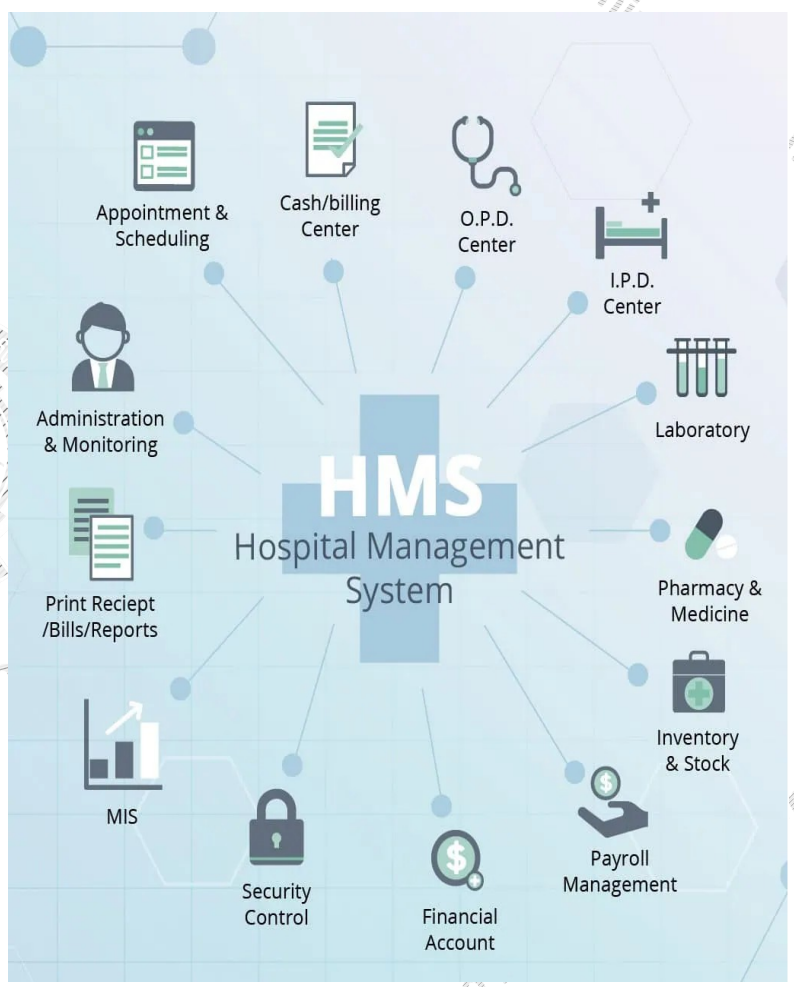
BITS Pilani

Modes of Data Flow - Through Databases

Example

- Data exchange through databases, can you share an example which comes to your mind immediately?
- Hospital management system
- These days all hospitals are automated for information flow
- Have you seen these days though doctors give a printout, in the pharmacy the medicines are distributed based on the online prescription?

- The application used by doctor and the pharmacist are different. So how the data is transferred or flown from one application to other?



Illustration

Doctor

Doctor's application

HMS DB

Patient's Mobile app

Pharmacist's application

Pharmacist

Patient visits doctor

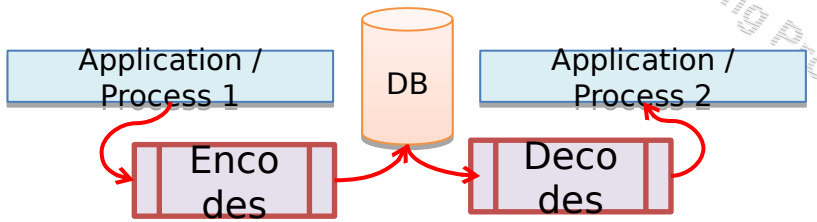
Patient

Gets Medicine

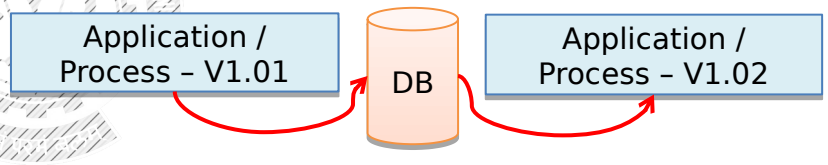
Additionally view the doctor's prescription on the mobile app

Scenarios

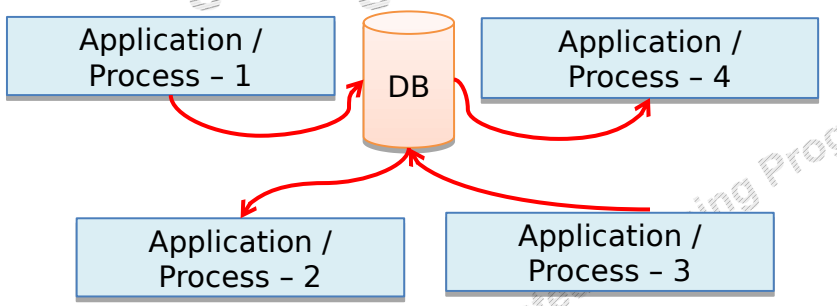
Scenario 1: the process that writes to the database encodes the data, and the process that reads from the database decodes it



Scenario 2: a single process accessing the database, in which case the reader is simply a later version of the same process. Backward compatibility is clearly necessary here, otherwise won't be able to decode what was previously wrote



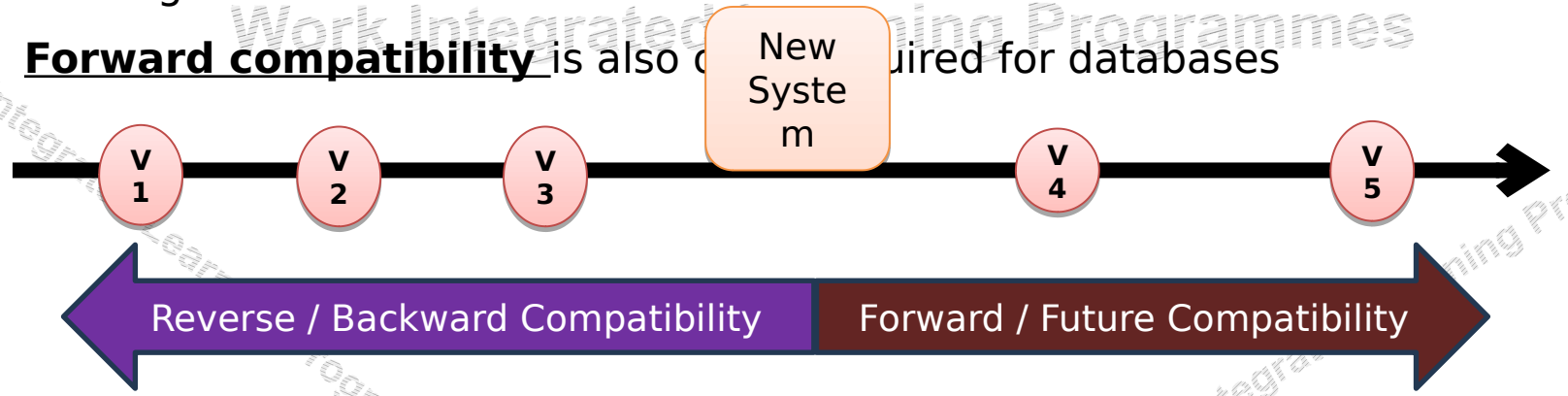
Scenario 3: several different processes to be accessing a database at the same time. For example, several different applications or services, or several instances of the same service



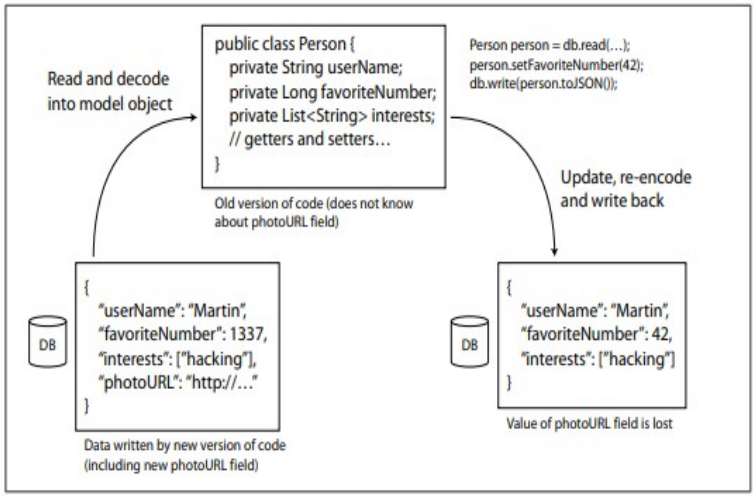
Points to Remember

In a changing environment

- Likely that some processes accessing the database will be running newer code and some will be running older code
- A new version is currently being deployed in a rolling upgrade, so some instances have been updated, while others haven't yet!
- Means that a value in the database may be written by a newer version of the code, and subsequently read by an older version of the code that is still running
- **Forward compatibility** is also c uired for databases



Changes to data



When an older version of the application updates data previously written by a newer version of the application, data may be lost if you're not careful.

- If a field is added to a record schema, and the newer code writes a value for that new field to the database. Subsequently, an older version of the code (which doesn't yet know about the new field) reads the record, updates it, and writes it back
- In this situation, the desirable behavior is usually for the old code to keep the new field intact, even though it couldn't be interpreted
- The encoding formats support such preservation of unknown fields, but sometimes need to take care at an

Points to Remember

- Different values written at different times
  - A database generally allows any value to be updated at any time
  - Means within a single database may have some values that were written five milliseconds ago, and some values that were written five years ago
- Data outlives code
  - When deploy a new version of your application, may entirely replace the old version with the new version within a few minutes
  - Same is not true of database contents: the five-year-old data will still be there

Points to Remember

- Schema Evolution
  - Rewriting (migrating) data to a new schema is certainly possible, but it's an expensive thing to do on a large dataset, so mostly avoided
  - Simple schema changes, such as adding a new column with a null default value, without rewriting existing data
  - Schema evolution thus allows the entire database to appear as if it was encoded with a single schema, even though the underlying storage may contain records encoded with various historical versions of the schema
- Archival storage
  - Snapshot of database can be taken from time to time, say for backup purposes or for loading into a data warehouse
  - Data dump will typically be encoded using the latest schema, even if the original encoding in the source database contained a mixture of schema versions from different eras
  - Copying the data anyway, might as well encode the copy of the data consistently

Work Integrated Learning Programmes

innovate achieve lead

BITS Pilani

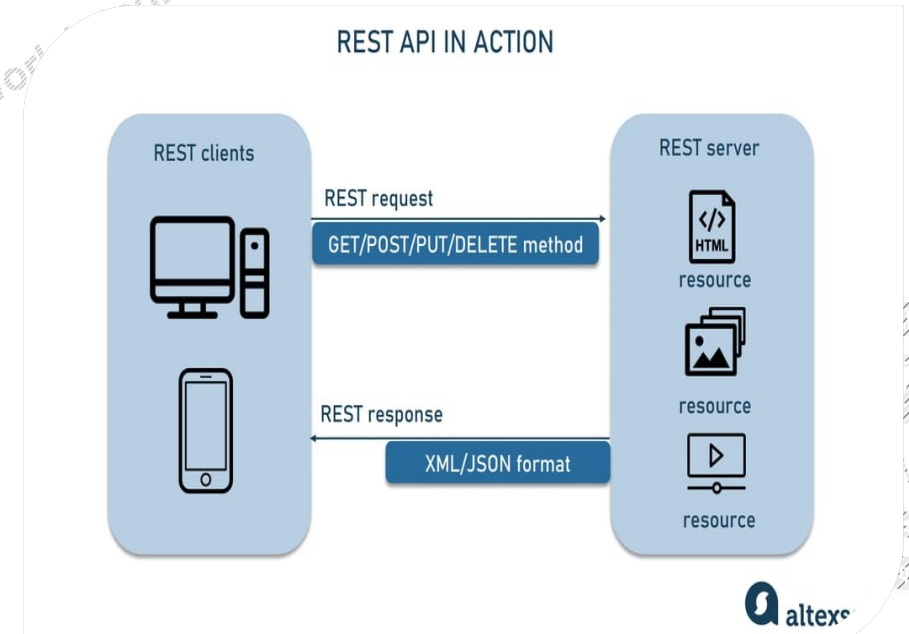
Modes of Data Flow - Through Services

Introduction

- Processes that need to communicate over a network, there are a few different ways of arranging that communication
- We are going to discuss about
  - Client Server Architecture
  - Service Oriented Architecture(SOA)
  - Microservices Architecture
  - Web Services
  - Remote Procedure Calls

innovate achieve lead
Work Integrated Learning Programmes

Client Server Architecture



- The servers expose an API over the network, and the clients can connect to the servers to make requests to that API
- The API exposed by the server is known as a service

Web Clients

- Clients (web browsers) make requests to web servers, making GET requests to download HTML, CSS, JavaScript, images etc. or making POST requests to submit data to the server
- The API consists of a standardized set of protocols and data formats (HTTP, URLs, SSL/TLS, HTML, etc.)
- Because web browsers, web servers and website authors mostly agree on these standards, can use any web browser to access any website

Work Integrated Learning Programmes

Client Server Architecture

innovate achieve lead

Other Clients

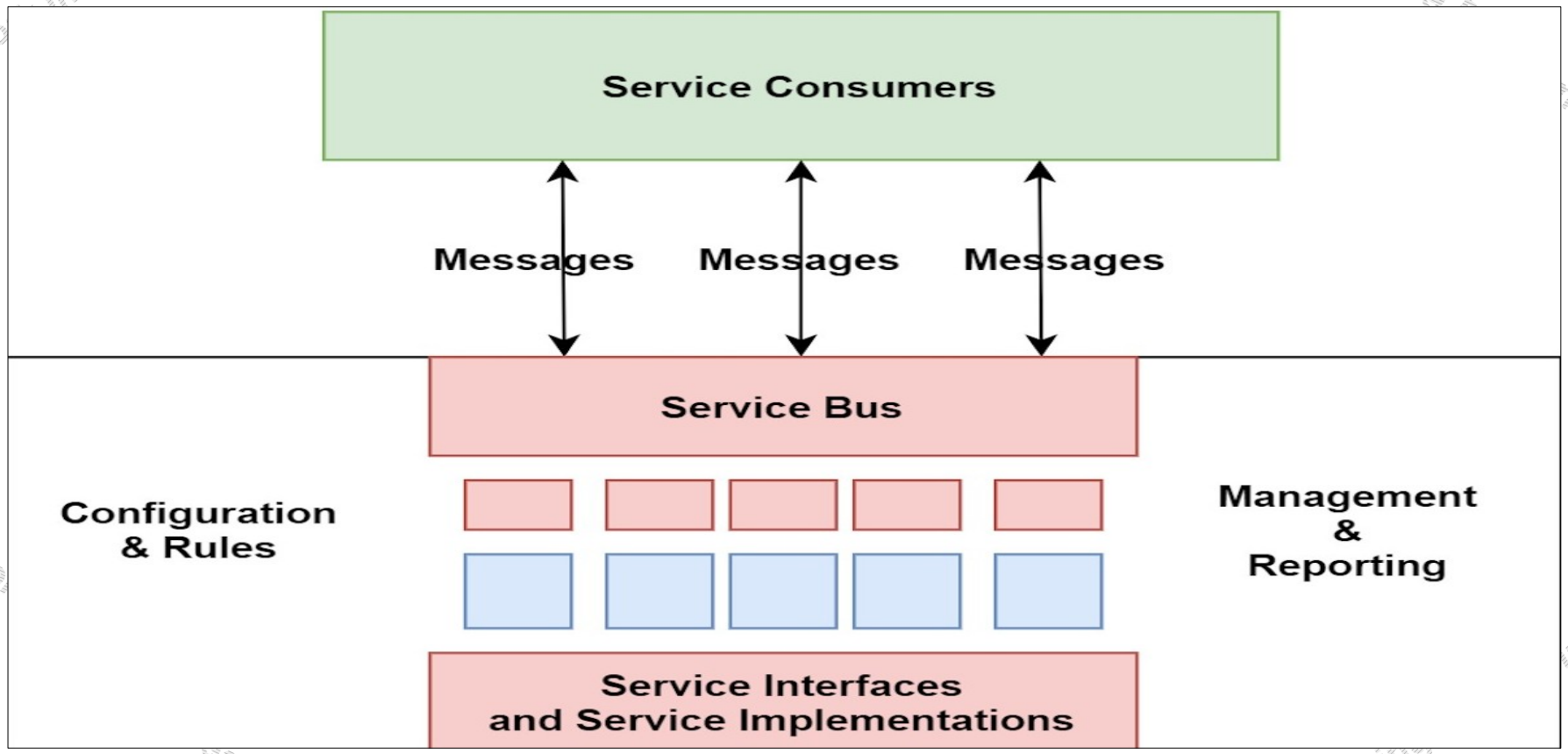
- A native app running on a mobile device or a desktop computer can also make network requests to a server
- A client-side JavaScript application running inside a web browser can use XMLHttpRequest to become a HTTP client (aka Ajax)
- The server's response is typically not HTML for displaying to a human, but rather data in an encoding that is convenient for further processing by the client-side application code (such as JSON)
- API is implemented on top of HTTP is application-specific Client and server need to agree on the details of that API

Work Integrated Learning Programmes

Services Oriented Architecture

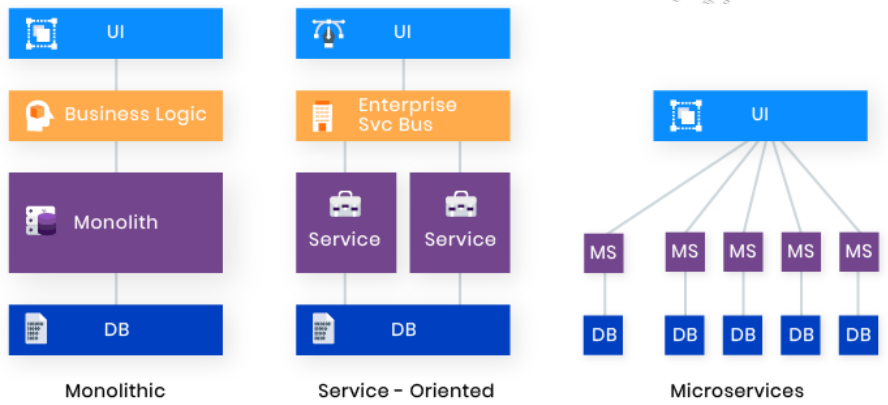
- Server can itself be a client to another service
- Often used to decompose a large application into smaller services by area of functionality, such that one service makes a request to another when it requires some functionality or data from that other service
- A key design goal is to make the application easier to change and maintain by making services independently deployable and evolvable. For example, each service should be owned by one team, and that team should be able to release new versions of the service frequently without having to coordinate with other teams
- Should expect old and new versions of servers and clients to be running at the same time
- Data encoding used by servers and clients must be compatible across versions of the service API

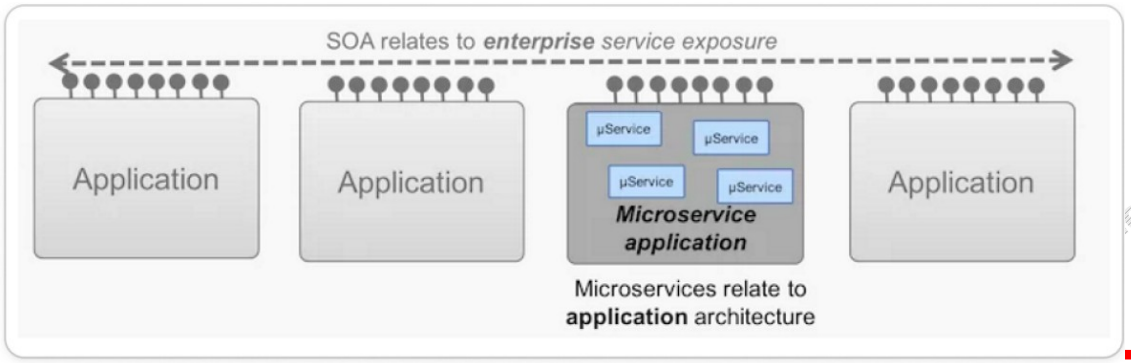
Services Oriented Architecture



SOA - Microservices

- Microservices is an architectural style evolved from SOA.
- Both are distributed architectures and both offer a decoupled paradigm.
- Whereas service-oriented architecure is heavier on infrastructure, microservices offers a more flexible, lightweight development style.





SOA vs Webservices

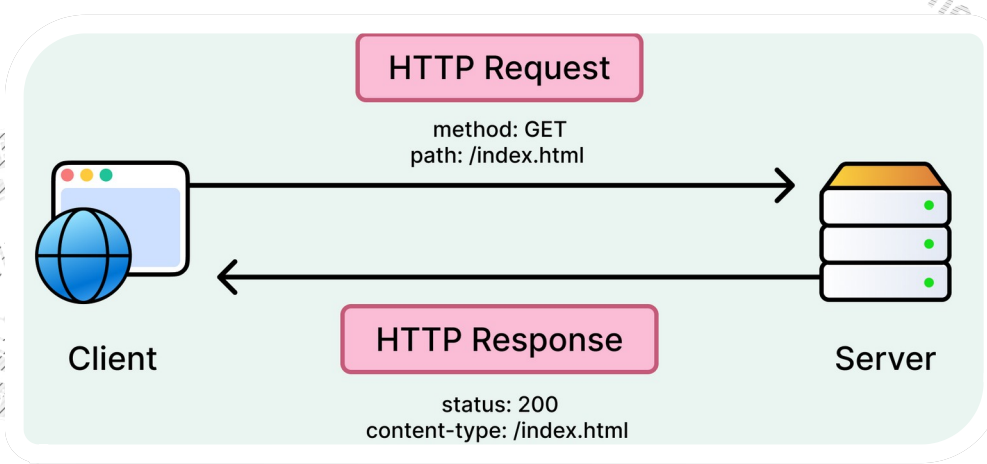
- SOA and web services are frequently conflated, but they are not the same thing.
- SOA is an architecture that allows developers to combine multiple application services into a larger, composite service.
- SOA can be implemented using SOAP-based web services or REST APIs, or sometimes a combination of both.
- It's important to understand that in SOA, a *service* is any remotely available resource that can respond to requests. A *web service* is implemented using specific protocols.

Work Integrated Learning Programmes

Webservices

HTTP:

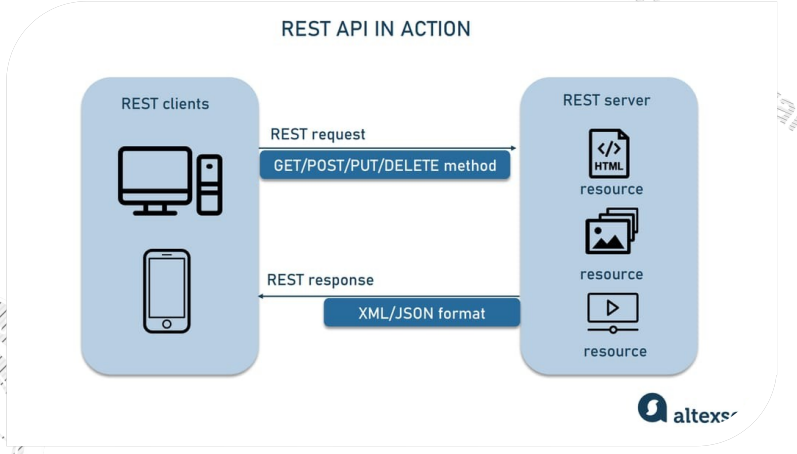
- HTTP, which stands for **Hypertext Transfer Protocol**, is the set of rules that allows web browsers to communicate with web servers to load webpages.
- When you visit a website, your browser sends an HTTP request to the server, which then sends back the data needed to display the page, such as text, images, and other files.



Work Integrated Learning Programmes

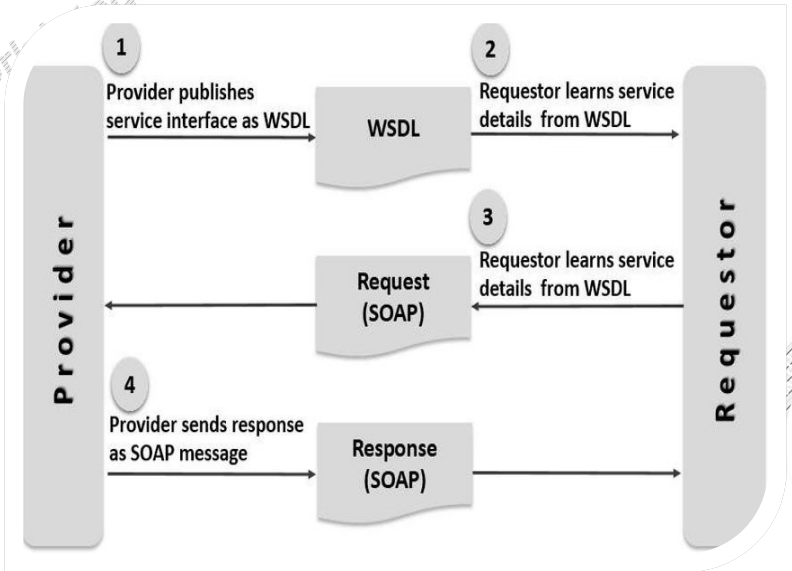
Webservices - REST

- Representational State Transfer
- REST is not a protocol, but rather a design philosophy that builds upon the principles of HTTP
- emphasizes simple data formats, using URLs for identifying resources, and using HTTP features for cache control, authentication, and content type negotiation
- gaining popularity compared to SOAP, at least in the context of cross-organizational service integration and is often associated with microservices
- An API designed according to the principles of REST is called RESTful



- good for experimentation and debugging (can simply make requests to it using a web browser or the command-line tool curl)
- supported by all main stream programming languages and platforms
- a vast ecosystem of tools(servers, caches, load balancers, proxies, firewalls, monitoring, debugging tools, testing)

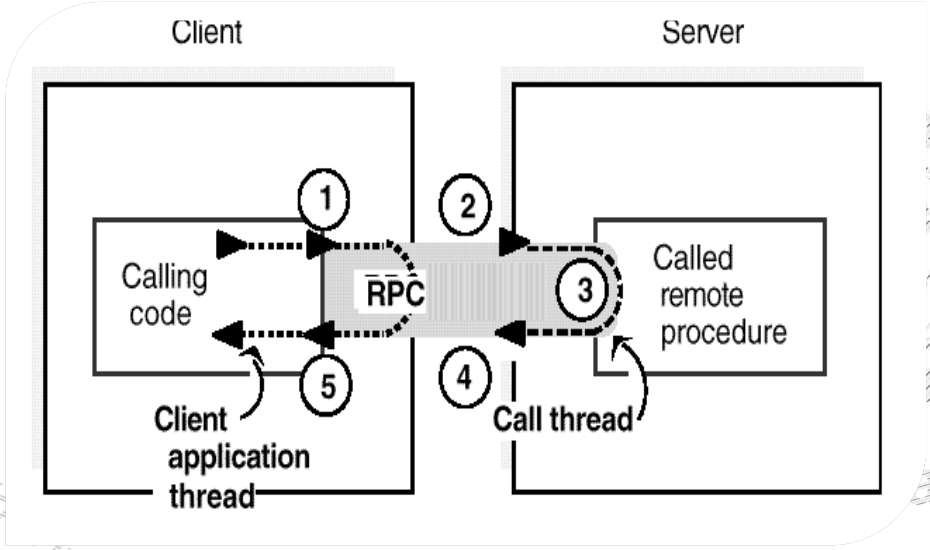
Webservices - SOAP



- Single Object Access Protocol
- SOAP is an XML-based protocol for making network API requests
- Most commonly used over HTTP, it aims to be independent from HTTP and avoids using most HTTP features

- Instead comes with a sprawling and complex multitude of related standards (the web service framework, known as WS-*)
- The API of a SOAP web service is described using an XML-based language called WSDL
- As WSDL is not designed to be human-readable, and as SOAP messages are often too complex to construct manually, users of SOAP rely heavily on tool support, code generation and IDEs
- Although SOAP is still used in many large enterprises, it has fallen out of favor in most smaller companies.

Remote Procedure Calls



- RPC tries to make a request to a remote network service look the same as calling a function or method in programming language, within the same process (aka location transparency)
- Although this seems convenient at first, the approach is fundamentally flawed . No point trying to make a remote service look too much like a local object in programming language, because it's a fundamentally different thing

Remote Procedure Calls – Current Directions

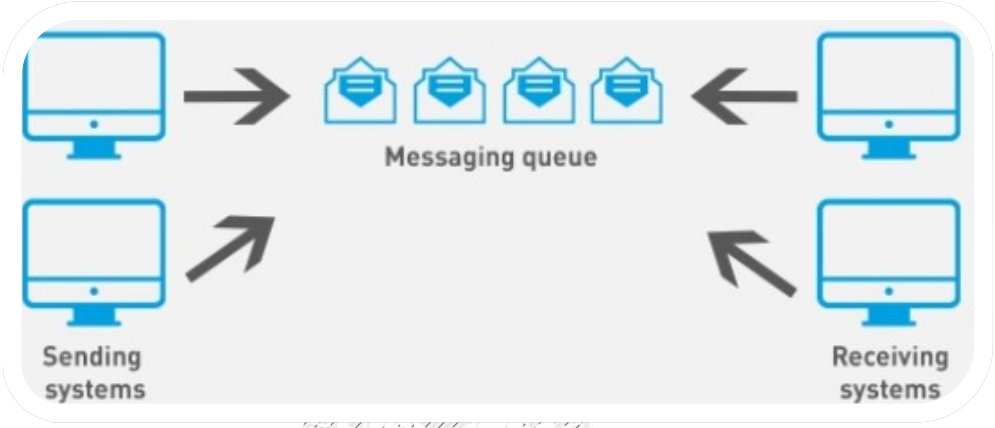
- RPC isn't going away!
  - Various RPC frameworks have been built on top of all the encodings:
  - Thrift and Avro come with RPC support included
  - gRPC is a RPC implementation using Protocol Buffers
- New generation of RPC frameworks is more explicit about the fact that a remote request is different from a local function call
  - Also provide service discovery — that is, allowing a client to find out at which IP address and port number it can find a particular service
  - Custom RPC protocols with a binary encoding format can achieve better performance than something generic like JSON over REST
  - Focus of RPC frameworks is on requests between services owned by the same organization, typically within the same datacenter

innovate achieve lead

BITS Pilani

Modes of Data Flow - Through Message Passing

Introduction



- Asynchronous message-passing systems, which are somewhere between RPC and databases
- similar to RPC in that a client's request (usually called a message) is delivered to another process with low latency
- similar to databases in that the message is not sent via a direct network connection, but goes via an intermediary called a message broker which stores the message temporarily
- also called a message queue or message-oriented middleware

Message Brokers

innovate achieve lead

Landscape

- In past, dominated by commercial enterprise software by companies such as TIBCO, IBM WebSphere, and webMethods.
- More recently, open source implementations such as RabbitMQ, ActiveMQ, HornetQ, NATS and Apache Kafka have become popular

Working

- one process sends a message to a named queue or topic
- broker ensures that the message is delivered to one or more consumers or subscribers of that queue or topic
- can be many producers and many consumers on the same topic
- Message brokers typically don't enforce any particular data model
- message is just a sequence of bytes, with some metadata, so can use any encoding format
- If the encoding is backward and forward compatible, have the greatest flexibility to change publishers and consumers independently and deploy them in any order

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Message broker over direct RPC

- can act as a buffer if the recipient is unavailable or overloaded, and thus improve system reliability
- can automatically redeliver messages to a process that crashed, and thus prevent messages from being lost
- avoids the sender needing to know the IP address and port number of the recipient
- allows one message to be sent to several recipients
- logically decouples the sender from the recipient (the sender just publishes messages and doesn't care who consumes them)
- Message-passing communication is usually one-way
  - sender normally doesn't expect to receive a reply to its messages
  - possible for a process to send a response, but this would usually be done on a separate channel
- This is what makes it asynchronous: the sender doesn't wait for the message to be delivered, but simply sends it and then forgets about it

Work Integrated Learning Programmes
BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Modern Data Stack

Traditional/Legacy Data Stack(TDS) -Problems

innovate achieve lead

Long turn-around time to untangle and set up infrastructure

- Companies making use of on-premise infrastructure are responsible for all the costs associated with it
- The army of engineers required to keep everything maintained and running smoothly.
- As setup is so deeply interconnected, what may seem like a minor change might break other parts of the system
- Finding the exact logical coupling between the systems requires a lot of work-hours to analyze before any improvements can be made to the existing landscape

Expensive journey to insights

- A lot of the report generation is manually done, especially when the data is coming from different sources
- The report is manually generated, manually cleaned, and manually transferred to Excel
- Leads to errors being made, time being taken away from other business-critical tasks, and the inability to scale.

Work Integrated Learning Programmes

Traditional/Legacy Data Stack – Problems - contd

Slow response to new information

- As the company grows, so does its data and computational power needs
- Very costly in terms of resources and time when it comes to scaling out (expanding) on-premises infrastructure
- leads to a limit in how much computational power there is to analyze data
- data pipelines can take hours to complete, a problem that is compounded as the organization grows
- A TDS requires slow ETL (Extract, Transform, Load) operations before newly ingested data can conform to the rest of the data model
- A new data update can take weeks and many hours of refactoring before insights appear
- By the time the data is ready, the organization is unable to act in time resulting in missed opportunities.

Summary

rely on rigid, on-premises infrastructure that can struggle with scalability, flexibility and real-time processing.

What is Modern Data Stack?

- refers to integrated, cloud-based tools and technologies that enable the collection , ingestion , storage , cleaning, transformation, analysis and governance of data
- offers a cloud-native, modular solution designed to streamline automation , optimize costs and accelerate insights
- empowers self-service analytics and artificial intelligence (AI) applications that many companies rely on today.
- Think of MDS as the assembly line of digital operations—each component plays a role in moving data seamlessly from collection to analysis
- By automating and scaling workflows, MDS ensures that organizations can process, store and use data with precision, driving better decision-making and innovation.

TDS vs MDS

	Traditional Data Stack	Modern Data Stack
Infrastructure	relies on physical servers	cloud native
Scalability	requires manual scaling	scales dynamically with demand
Integration	relies on custom workflows	automates data ingestion
Flexibility	monolithic	modular, enabling seamless tool integration
Analytics	supports batch reporting	offers real-time insights and interactive dashboards.
Cost	involves significant upfront investment	uses pay-as-you-go models.

MDS - Key Characteristics

Enable self-service for a diverse range of users

- Easily discover and analyze data within the platform
- Understand the context associated with data, such as column descriptions, history and lineage
- Derive insights from data with minimal dependencies on the data or IT team

Enable "agile" data management

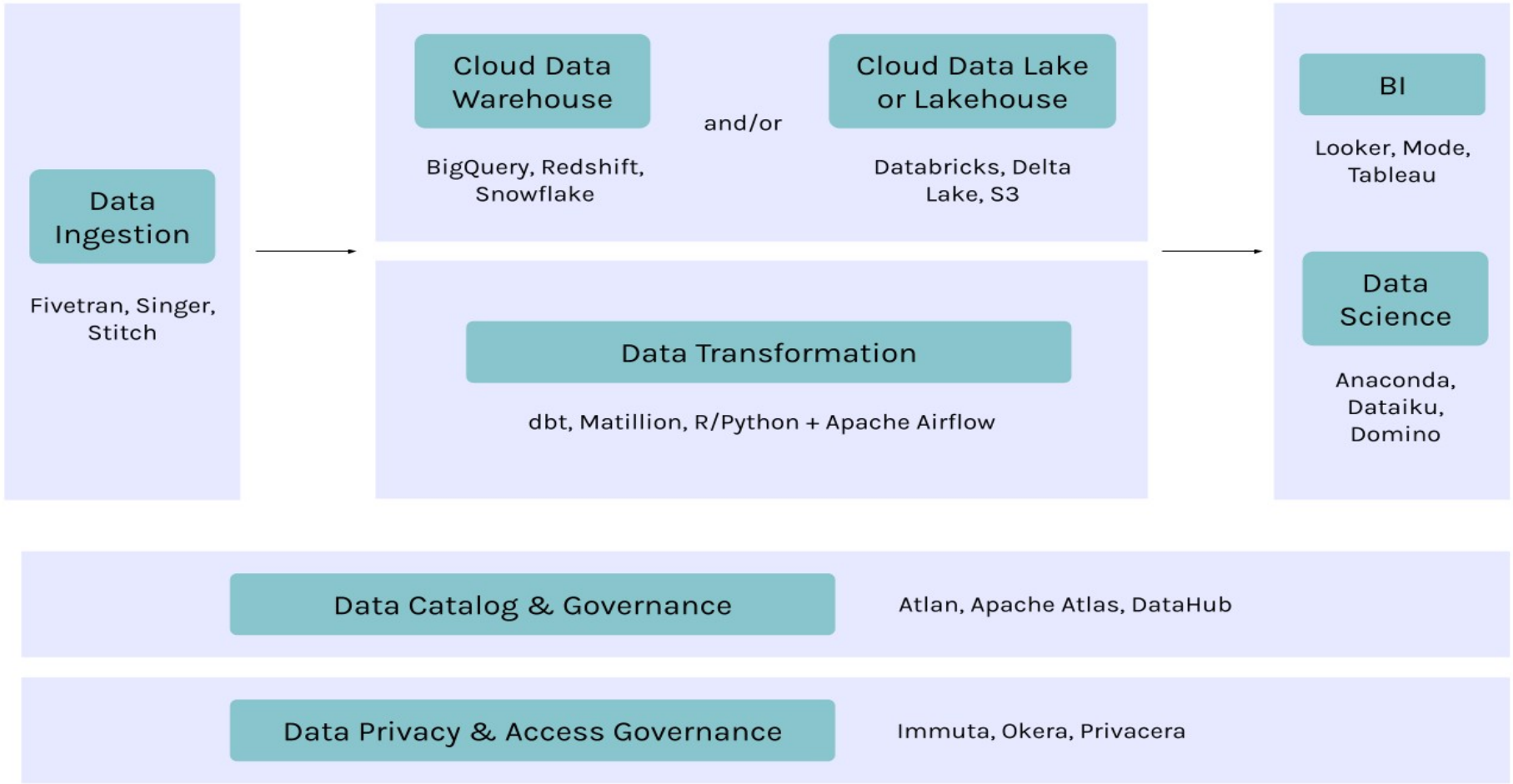
- *Availability*: Data is already available in a data lake or warehouse. Modern data lakes and warehouses separate storage and compute, which makes it possible to store large amounts of data for relatively cheap.
- *Elasticity*: Compute is based on a cloud platform, which allows for elasticity and auto-scalability. For example, if end users consume the most data and analytics on Friday afternoons, it should be possible to auto-scale processing power on Friday afternoon to give users a great experience and then scale down within a few hours.

MDS - Key Characteristics - contd

Flexible, fast set-up, and pay-as-you-go

- ***Easy to set up*** – no lengthy sales process, demo calls, and implementation cycles. Just login, pay via credit card, and go!
- ***Pay as you go*** – no up-front payments and million dollar licensing fees. The "modern" stack is all about putting power in the hands of the consumer, i.e. paying only for what you use
- ***Plug and play*** – the modern data stack will continue to evolve and innovate, and the best of breed tools do not enforce "lock in" like tools in the legacy era, and are instead built on open standards & APIs allowing easy integration with the rest of the stack

Key building blocks



Modern data tools

innovate achieve lead

Modern Data Ingestion

- Data ingestion is likely where your efforts to build out a modern data platform get started
- i.e. how do you bring data from a variety of different data sources and ingest it into your core data storage layer?
- The majority of data pipelines capture data in batch, using SQL queries or extract routines, but this is changing
- Many companies now prefer a streaming-first architecture that captures and processes data in real time in a continuous stream
- Streaming tools use change data capture, replication, and/or publish/subscribe messaging systems to capture events as they happen, move them into the data environment, and analyze them.

Modern data tools - contd

innovate achieve lead

Modern data storage and processing

The data storage and processing layer is fundamental to the modern data platform. While this architecture is evolving, we typically see 3 kinds of tools or frameworks:

Data warehouses

- These are generally the system of choice for analysts since they optimize compute and processing speed.

Data lakes

- A data lake architecture refers to data being stored on an object storage, coupled with tools to process the data
- These cheap storage systems are often used to store vast amounts of raw or even unstructured data.

Data lakehouses

- convergence of data warehouses and lakes
- Helps to unify the siloed systems
- a system design that combines the data management features such as ACID transactions and change data capture from a data warehouse with the low-cost storage of a data lake

Modern data tools - contd

innovate achieve lead

Modern data transformation - Core implementations

1. Native SQL

   For companies with data warehouse–first architectures, tools that leverage native SQL for transformation have emerged as the top choice for data transformation.

2. orchestration engine

   orchestration engine coupled with custom transformation in a programming language

Modern business intelligence and analytics

- BI dashboards have been around for ages, but the latest breed of BI and analytics tools are built to fit within a larger modern data platform.
- These are generally more self-service platforms that allow users to explore data, rather than just consuming graphs and charts

Work Integrated Learning Programmes

Modern data tools - contd

innovate achieve lead

Modern data catalogs and governance

- While the modern data platform is great in some areas (super fast, easy to scale up, little overhead), it struggles with bringing discovery, trust and context to data.
- As metadata itself becomes big data, we're in the midst of a leap forward in metadata management

Modern data privacy and access governance

- a major challenge is to be able to manage privacy controls and access governance across your entire stack.
- tools that can act as an entitlement engine to apply privacy and security policies across the data stack.

Other tools

- Real-time data processing tools
- Data science tools
- Event collectors
- Data quality tools

Work Integrated Learning Programmes

Modern data tools – Summary

Component	Tools
data ingestion	▪ SAAS tools: Fivetran, Hevo Data, Stitch ▪ Open-source tools: Singer, StreamSets ▪ Custom data ingestion pipelines built on orchestration engines like Airflow
data storage and processing	▪ **Data warehouses :** BigQuery, Redshift, , Snowflake. ▪ **Data lakes :** ❑ Data storage: Amazon S3, Azure Data Lake Storage Gen2, Google Cloud Storage ❑ Data processing engines: Databricks or Spark; Athena, Presto, or Starburst; Dremio ❑ **Data LakeHouse**: Databricks, Snowflake, and Google BigLake, Delta Lake, Apache Hudi, Apache Iceberg
data transformation	▪ With data warehouses: dbt, Matillion ▪ With an orchestration engine: Apache Airflow + Python, R, or SQL
business intelligence and analytics	Looker, Mode, Redash, Sigma, Sisense, Superset, and Tableau.

Work Integrated Learning Programmes

Modern data tools – Summary - Contd

Component	Tools
data catalogs and governance	▪ SAAS tools: Atlan ▪ Open-source tools: Apache Atlas, LinkedIn's DataHub, Lyft's Amundsen ▪ In-house tools: Airbnb's Dataportal, Facebook's Nemo, Uber's Databook
data privacy and access governance	▪ SAAS services: Immuta, Okera, Privacera ▪ Open-source engines: Apache Ranger

Work Integrated Learning Programmes

Modern data tools – Summary - contd

Component	Tools
Real-time data processing	▪ Real-time streaming pipelines: Confluent, Kafka ▪ Real-time analytics: Druid, Imply
Data science	▪ Preferred by data scientists: Jupyter Notebooks ▪ Other options: Dataiku, DataRobot, Domino, SageMaker
Event collectors	▪ Segment , Snowplow
Data quality	▪ Data profiling: This is getting engulfed by data catalog and governance tools like Atlan or open source profiling frameworks like Amazon Deequ. ▪ Unit testing: Frameworks like the open-source Great Expectations

Work Integrated Learning Programmes

Use Cases

AI-powered personalization

MDS allows businesses to deliver data-driven AI personalization. This personalization can help optimize user experiences across areas such as e-commerce, streaming platforms and software as a service (SaaS) applications. Using Apache Spark for real-time processing and Databricks for scalable analytics, data scientists can analyze customer preferences and engagement to improve recommendation engines and content delivery networks.

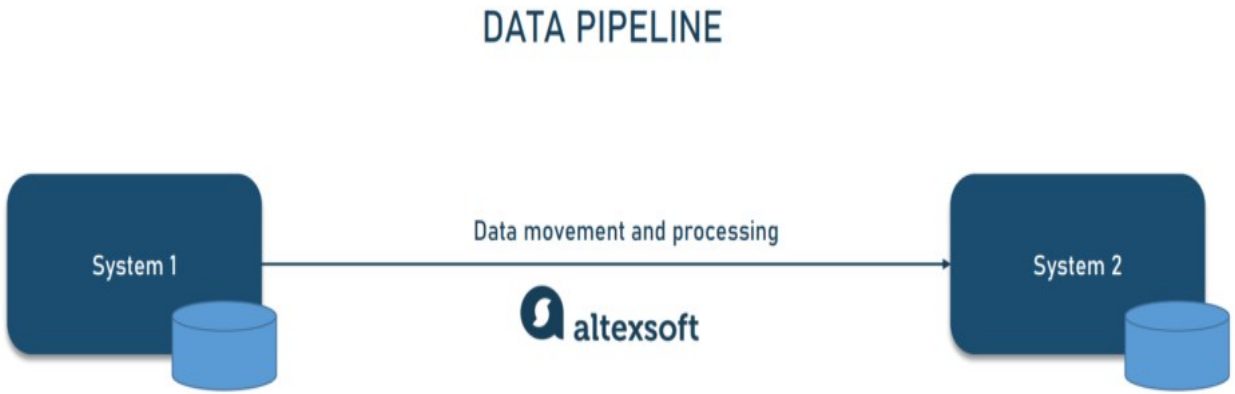
Customer insights and sales optimization

Companies use MDS and SaaS analytics tools to track customer behavior and refine marketing strategies. Cloud platforms such as Snowflake and Looker generate real-time dashboards for categories such as purchase patterns and pricing optimization—all of which can help businesses boost conversion rates and retention.

Data Pipeline

Re-Introduction

- Series of steps that allow data from one system to move to become useful in another system particularly analytics, data science, or AI and machine learning systems
- Works by pulling data from the source, applying rules for transformation and processing, then pushing data to its destination.



Origin

- Emerged from the big data community

- Data lakes created a feeding frenzy among developers, data scientists, and data analysts
  - gained unfettered access for the first time
  - to a nearly limitless supply of raw data sourced
  - from a multiplicity of internal and external systems

- Created a wide diversity of workflows and programs—or data pipelines—to process that data
  - (e.g., clean, filter, aggregate, move, load)
  - for a variety of purposes and use cases

Work Integrated Learning Programmes

Situation

- Though data is a critical business asset most organizations still don't have mature processes for converting data into insights that drive business value
- The development of data and analytics pipelines, is still a handcrafted and largely non-repeatable process with minimal reuse, managed by individuals working in isolation with different tools and approaches
- The result is
  - a plodding development environment that can't keep pace with the demands of a data-driven business
  - an error-prone operational environment that is slow to respond to change requests
- Drawing Swords
  - The lack of efficiency on the development side leads to a lack of effectiveness on the business side

Purpose

- Lot of data out there!
- Data pipelines transform raw data into data ready for analytics, applications, machine learning and AI systems
  - keep data flowing to solve problems, inform decisions
- Used to:
  - Deliver sales data to sales and marketing for customer 360
  - Link a global network of scientists and doctors to speed drug discovery
  - Recommend financial services to help a small business owner thrive
  - Track COVID-19 cases and inform community health decisions
  - Combine diverse sensor data with AI for predictive maintenance
- With so much work to do, data pipelines can get pretty complicated pretty fast.

Benefits

Self-service Data

- Can be created ad hoc by data scientists and business analysts unstick the IT bottleneck
- Means when people have brilliant ideas, they can test them, fail fast, and innovate faster

Real-time Analytics and Applications

- Real-time or near real-time functionality in consumer and business applications puts the pressure on data pipelines to deliver the right data, to the right place, right now
- Streaming data pipelines deliver continuous data to real-time analytics and applications

Accelerate Cloud Migration and Adoption

- Helps to expand cloud presence and migrate data to cloud platforms
- Cloud computing helps to feed many new use cases at processing speeds, cost-effectiveness, and bursting capacity unheard of in traditional on-premises data centers
- Teams can take advantage of rapid innovation happening on those cloud platforms such as
  - natural language processing
  - sentiment analysis
  - image processing

Challenges

Always Under Construction Data

- Building and debugging data pipelines takes time
- Have to align with the schema, set sources and destinations, check work, find errors, and back and forth until one can finally go live
- by which time the business requirements may have changed again
- That's why many data engineers have such a backlog of work

Out of Order Data Pipelines

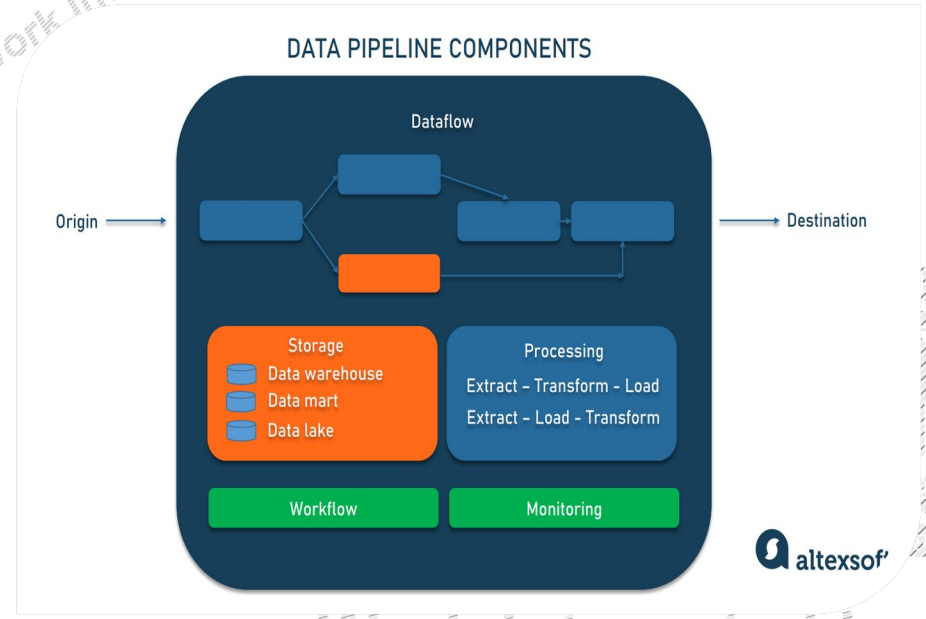
- Even a small change to a row or a table can mean hours of rework, updating each stage in the pipeline, debugging, and then deploying the new data pipeline
- Data pipelines often have to go offline to make updates or fixes
- Unplanned changes can cause hidden breakages that take months of engineering time to uncover and fix
- These unexpected, unplanned, and unrelenting changes are referred to as “data drift”.

Build It and They Will Come

- Data pipelines are built for specific frameworks, processors, and platforms.
- Changing any one of those infrastructure technologies to take advantage of cost savings or other optimizations can mean weeks or months of rebuilding and testing pipelines before deployment

Data Pipeline Architecture

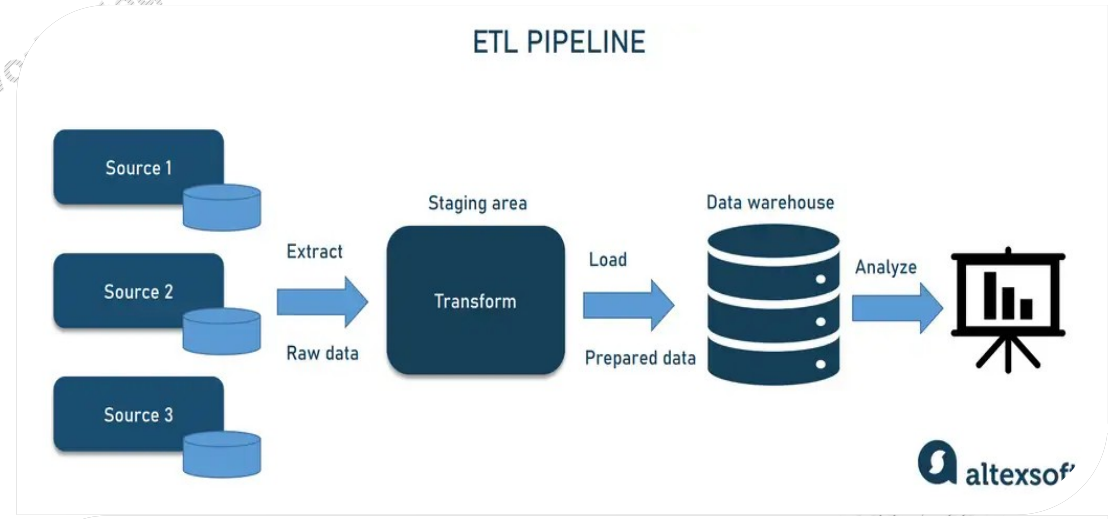
Components

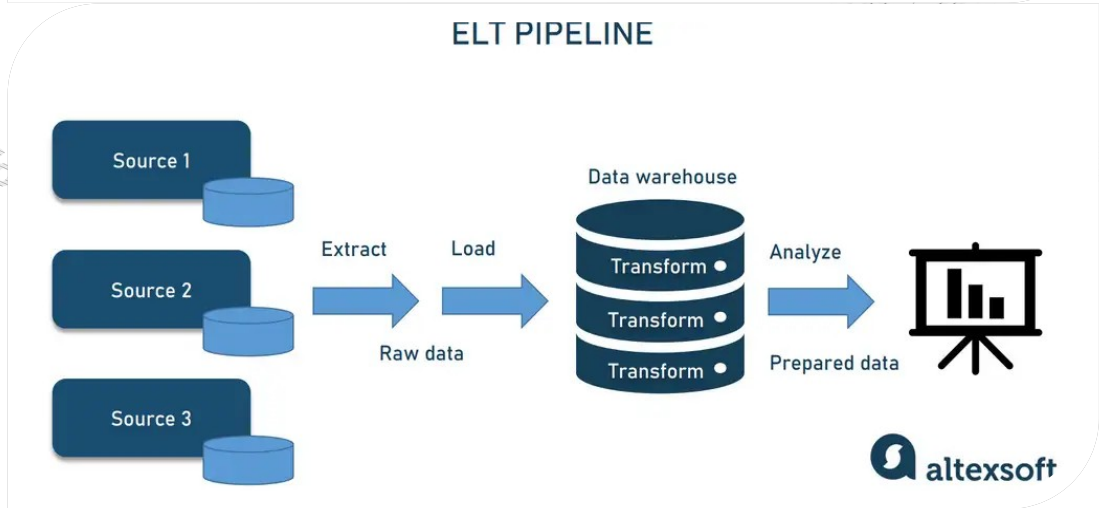


- Origin is the point of data entry in a data pipeline.
- The final point to which data is transferred is called a destination.

- That's the movement of data from origin to destination, including the changes it undergoes along the way as well as the data stores it goes through.
- Storage refers to systems where data is preserved at different stages as it moves through the pipeline.
- Processing includes activities and steps for ingesting data from sources, storing it, transforming it, and delivering it to a destination.
- The workflow defines a sequence of processes (tasks) and their dependence on each other in a data pipeline. The goal of monitoring is to check how the data pipeline and its stages are working

ETL, ELT





- Both ETL and ELT are a type of data pipeline architecture

ETL, ELT

- The target (e.g., cloud data platform) can handle massive parallel transformations
- You want to load raw data quickly for agility
- You work with semi-structured / unstructured data
- You need data reusability for multiple downstream consumer

Eg:

Data Lake / Lakehouse Pipelines

Machine Learning Pipelines

ELT

- You need high data quality and governance upfront
- The target system (e.g., legacy warehouse) has limited compute/storage
- You need structured, curated, analytical data
- Transformations are complex or depend on utside the database

Eg:

Enterprise Data warehouse

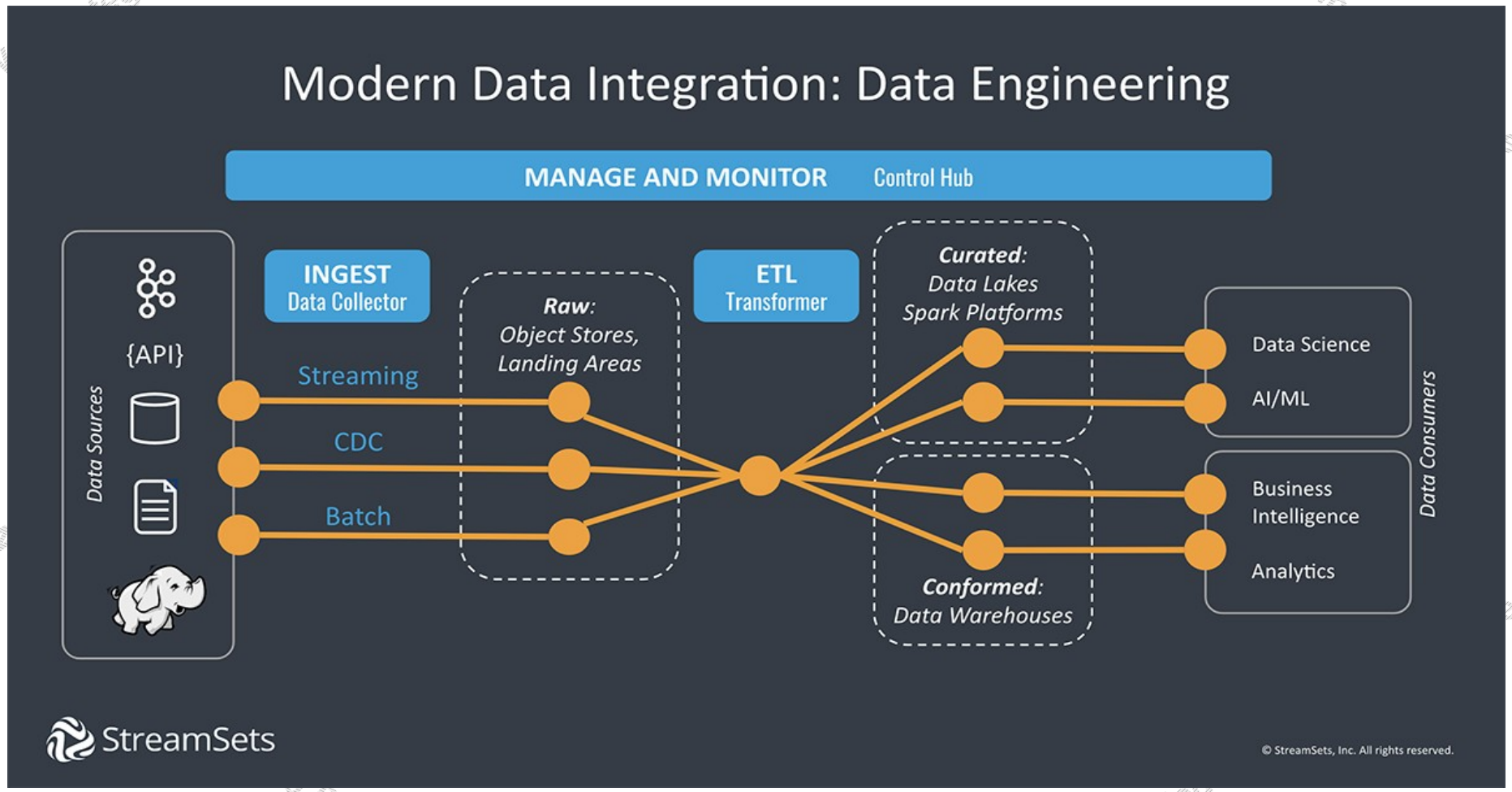
Regulatory / Compliance Reporting

Master Data Management

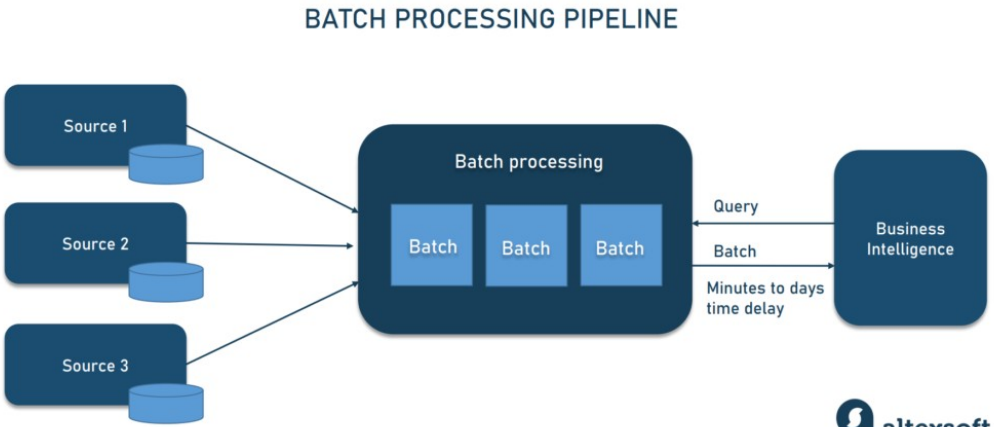
ETL

Which is ETL and which is ELT?

Data Integration Pipeline



Batch Processing Pipeline

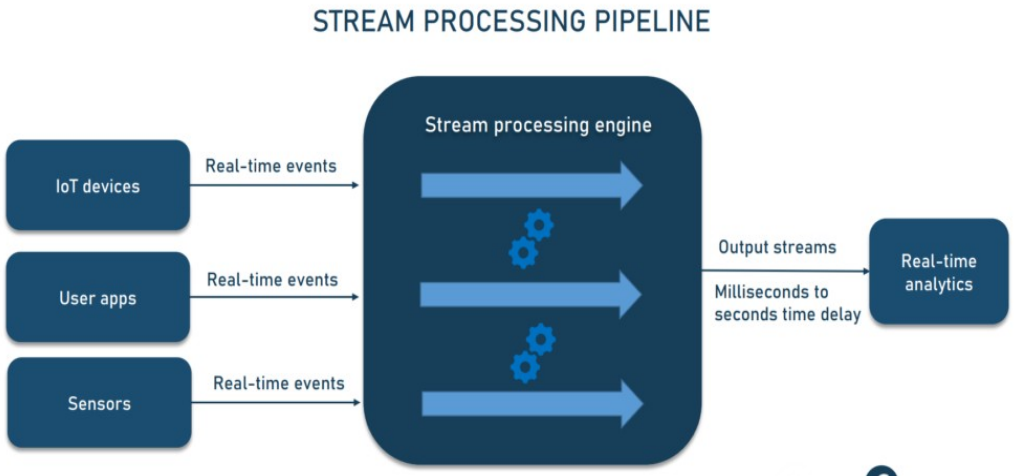


- Loads "batches" of data into a repository during set time intervals, which are typically scheduled during off-peak business hours.
- the optimal data pipeline when there isn't an immediate need to analyze a specific dataset (for example, monthly accounting), and it is more associated with the ETL data integration process
- Batch processing jobs form a workflow of sequenced commands, where the output of one command becomes the input of the next command.

- The development of batch processing was a critical step in building data infrastructures that were reliable and scalable.

Work Integrated Learning Programmes

Stream Processing Pipelines



- Also known as event-driven architectures—continuously process events generated by various sources, such as sensors or user interactions within an application

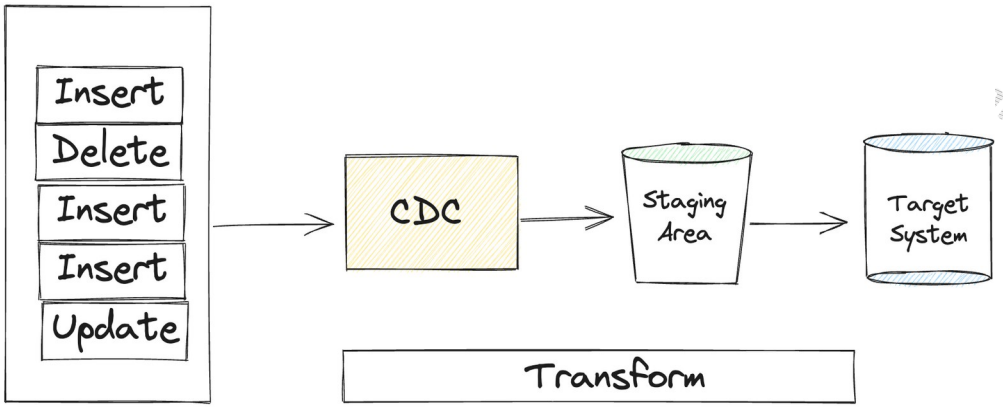
- Events are processed and analyzed, and then either stored in databases or sent downstream for further analysis.
- Streaming data is leveraged when it is required for data to be continuously updated.
- Since data events are processed shortly after occurring, streaming processing systems have lower latency than batch systems, but aren't considered as reliable as batch processing systems as messages can be unintentionally dropped or spend a long time in queue.

Work Integrated Learning Programmes

Change Data Capture(CDC)

- **Scenario:** An ecommerce backend service which talks about stocks and stock keeping units. This is continuously used whenever a product is searched from the front end UI(which could be web page or mobile app). These get updated as and when new stock arrive or on customer purchase or return . In case the database is down, they want to bring in the backup or replica database to run without any service interruption and with no or very minimum data loss. How will you achieve this?

Used in these scenarios



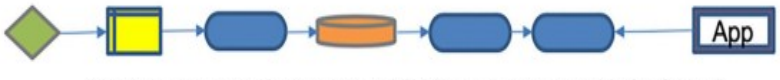
- A method for extracting each change event(insert, update, delete) that occurs in a DB
- RDBMS generates event logs, which can be used for replications
  (batch CDC)
- Many cloud based NoSQL, can send events (trigger CDC)

Dataflow Model & Unified batch & streaming

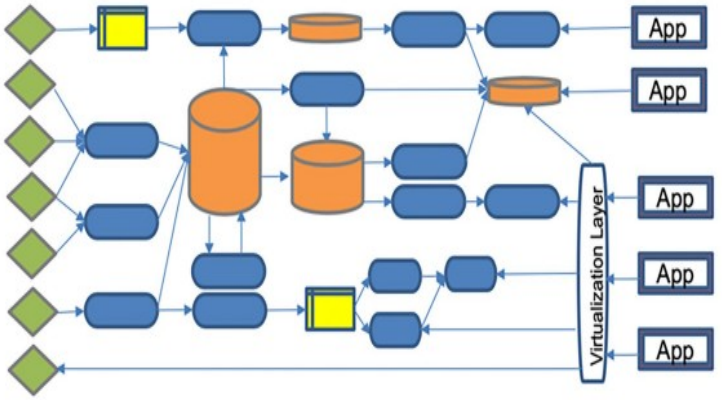
- One of the central problems of managing batch and stream processing unifying multiple code paths
- Kappa – relies on unified queueing and storage layer, but still has to confront using different tools for collecting real-time statistics or running batch aggregation jobs
- Google attempted to solve it by Dataflow model and Apache Beam as its implementation
  - Core idea is to view all data as events, aggregation is performed over various types of windows
  - Ongoing real-time event streams are unbounded data
  - Data batches are simply bounded event streams and boundaries provides natural window
  - Windows can tumbling , sliding
  - Real-time and batch processing happens in same system using nearly identical code
- Philosophy "batch as a special case of streaming" is now more pervasive

Data Pipeline Implementation

Simple and Complex



SIMPLE - "Export this data into a CSV file and place into this file folder."



COMPLEX - "Move tables from 10 sources into a target database, merge common fields, array into a dimensional schema, aggregate by year, flag null values, convert into an extract for a BI tool, and generate personalized dashboards based on the data."

- Some data pipelines are simple
- CSV → Database → Dashboard
  - Data from CSV files (sales or IoT logs) is ingested daily into a relational database, then visualized in a dashboard.
- Simple ETL, ELT

- Most are more complex
- Eg: Move tables from 10 sources into a target database, merge common fields, array into a dimensional schema, aggregate by year, flag null values, convert into an extract for a BI tool, and generate personalized dashboards based on the data.

Work Integrated Learning Programmes

Scenarios

A development pipeline

- creates code to process data for a new pipeline

An execution pipeline

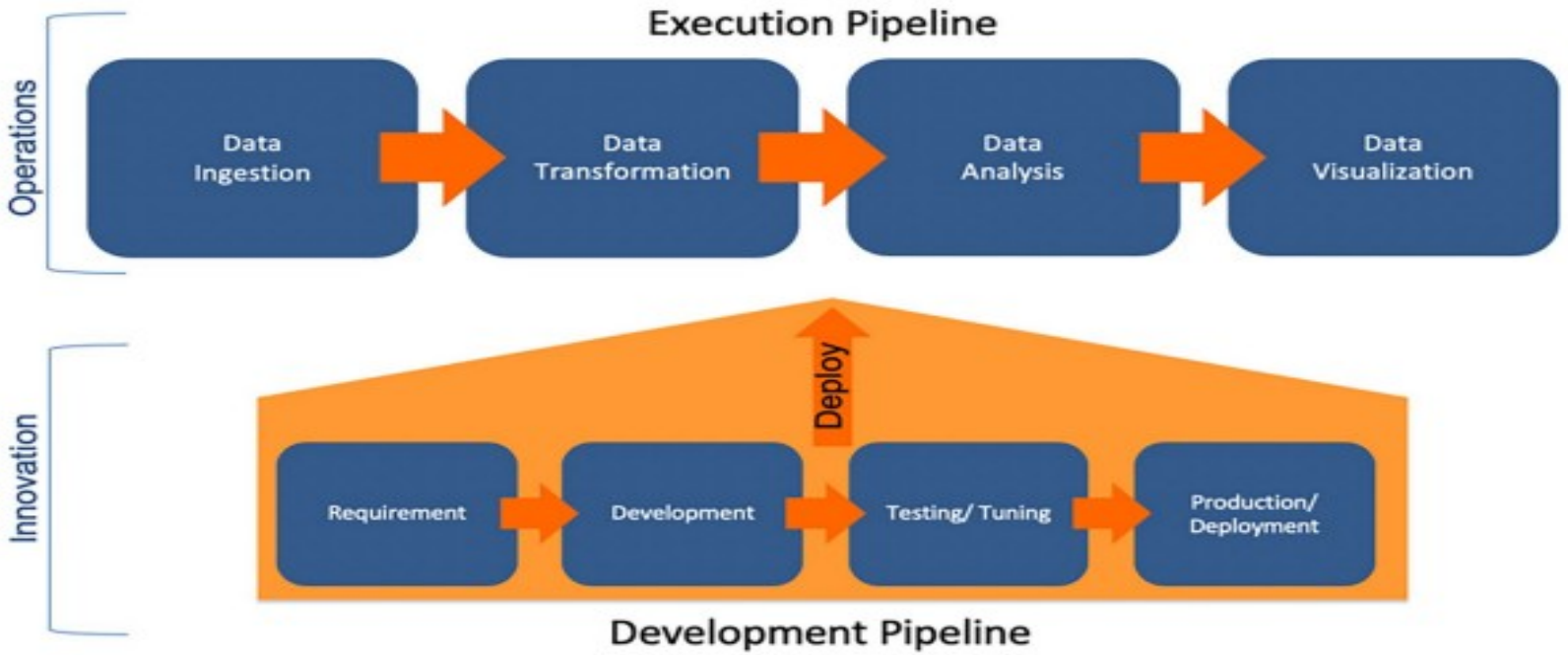
- runs the pipeline in production

In most IT shops

- these are managed by different teams, creating tremendous inefficiencies
- developers can build or change code without being responsible for its downstream impacts
- inevitably increases error rates, delays delivery and frustrates business users

Work Integrated Learning Programmes

Development and Execution Pipelines



Development and Operations Pipelines

Micro Pipeline

- To manage the complexity, it is better to divide individual stages into micro-pipelines
- to ensure that changes can be made to each step without disrupting the entire flow

The micro-pipelines consist of the following steps:

Requirement

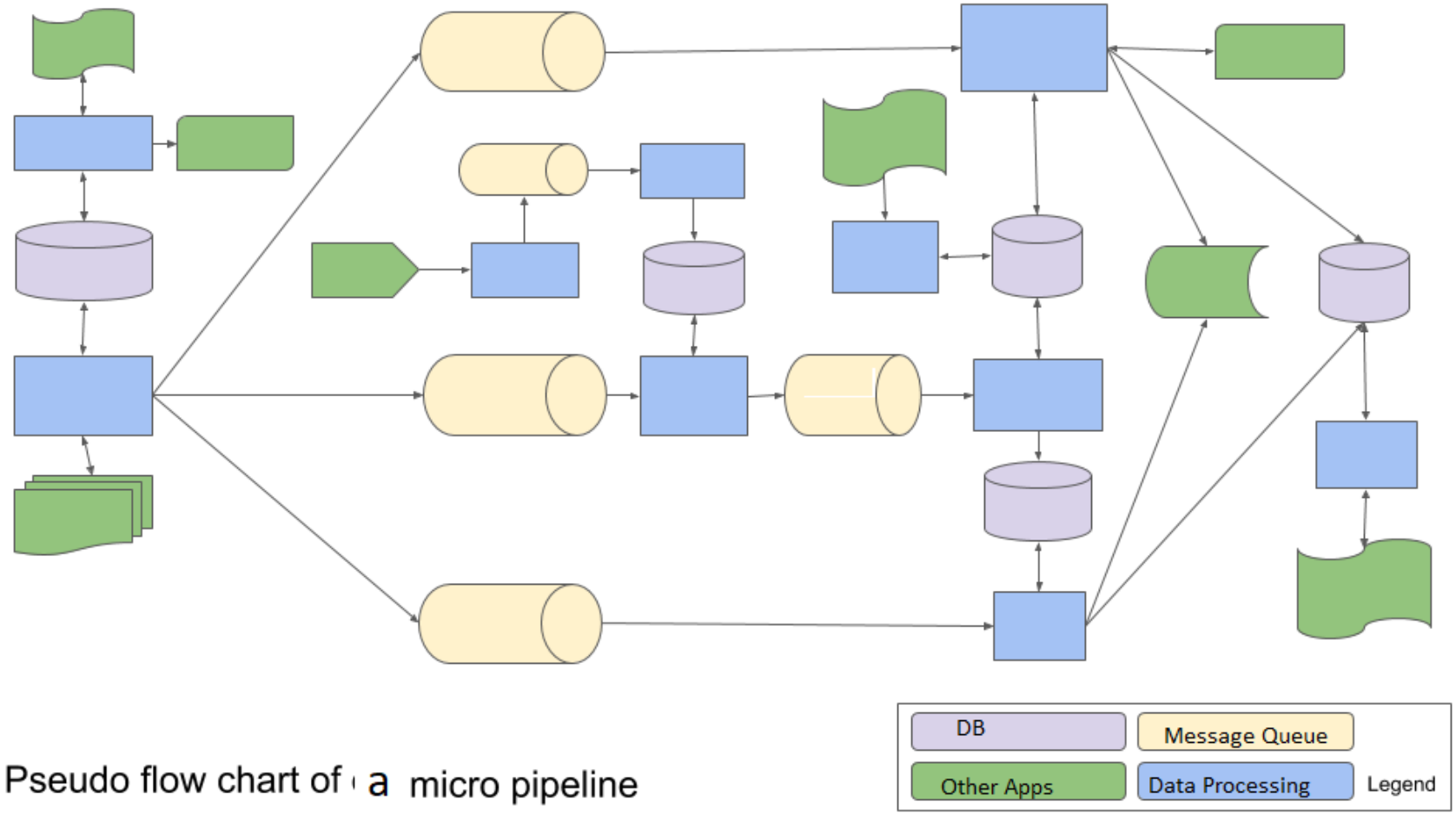
- Agile methods, such as Scrum and Kanban, elicit requirements from stakeholders in an iterative manner
- A requirement could be the integration of a new data source in an ETL process or the addition of a new field to a database table

Development

- In data and analytics, development does not necessarily mean programming
- it can comprise tasks like changing a data model or reconfiguring a dash-board
- Requires a high level of automation
- e.g., automatically generating database queries when a model is changed, to avoid repetitive manual work

Work Integrated Learning Programmes

Micro Pipeline - Sample



Pseudo flow chart of a micro pipeline

innovate achieve lead

BITS Pilani

Data Pipeline Tools / Platforms

Considerations - Overall

- Building a single pipeline for a single purpose at a given time is no problem
- Can grab a simple tool for setting up a data pipeline or hand code the steps
- But how to scale that process to thousands of data pipelines in support of increasing demand for data across organization, over months or years?
- When considering data pipeline tools, it is important to think ahead to where your data platform is headed.

Decision Basis

- Are we grabbing data from one place to put it somewhere else? Or do you need to transform it to fit the downstream analytics requirements?
- Is our data environment stable and fully under your control? Or is it dynamic and pulling data from systems or apps outside of your control?
- Will the pipeline move data once, for a short-term analysis project? Or will the pipelines you build need to be operationalized to handle data flows over time?

Work Integrated Learning Programmes

Data Ingestion and Data Loading Tools

- These tools make data pipelines easy to set up, easy to build, and easy to deploy solve the “under construction” issue
- but only if you can count on your data scientists to do the data prep work later
- Only support the most basic transformations and work best for simply copying data
- Instead of being intent-driven, these data pipelines are rigid and embedded with the specifics of data structure and semantics
- Instead of adapting to change when data drift happens, they have to be rebuilt and deployed again

Work Integrated Learning Programmes

Data Integration, Data Transformation Platforms

- More complex data integration or ETL software may have a solution for every possible scenario with hundreds of connectors, integrations, and transformations
- But these platforms were designed for an era when data drift hardly happened
- Things stayed the same for years
- At the first hint of change, these data pipelines break and require massive rework
- The demands for digital transformation are moving fast and planning for every possible outcome may not be possible

Data Engineering Platforms

A data engineering platform builds smart data pipelines according to DataOps principles

- Smart data pipelines abstract away the “how” so you can focus on the what, who, and where of the data
- This is the fundamental difference between data integration and data engineering

Instead of being perpetually under construction, out of order, or limited to a single platform

- smart data pipelines allow you to move fast with confidence that your data will continue to flow with little to no intervention

DataOps tools allow us to:

- Design and deploy data pipelines in hours, not weeks or months
- Build in as much resiliency as possible to handle changes
- Adopt to new platforms by pointing to them, a task that takes minutes not months

Work Integrated Learning Programmes

Use Cases-Contd

Logistics and supply chain optimization

By integrating Fivetran for data ingestion and dbt for transformation, companies can monitor inventory in real time and predict disruptions. This integration can lead to faster fulfillment, reduced costs and improved demand planning in industries such as retail, manufacturing and transportation.

Fraud detection and risk management

Financial institutions and e-commerce platforms use MDS to detect fraud and prevent data breaches. By using ML models, application programming interfaces (APIs) and services such as Amazon Redshift, organizations can identify suspicious transactions and automate fraud detection.

References

- Designing Data Intensive Applications by Martin Kleppmann
- https://www.ibm.com/think/topics/data-pipeline
- https://medium.com/@andymadson/the-evolution-of-data-pipelines-from-etl-to-elt-and-beyond-2e04deac7062
- https://www.infoworld.com/article/2158174/what-is-service-oriented-architecture.html
- https://www.altexsoft.com/blog/data-pipeline-components-and-types/
- https://underthehood.meltwater.com/blog/2019/07/12/micro-pipelines-analyzing-big-data-with-tiny-apps/
- https://www.ibm.com/think/topics/modern-data-stack

Work Integrated Learning Programmes

innovate achieve lead

BITS Pilani

Thank you